USF UNIVERSITY OF SAN FRANCISCO

CHANGE THE WORLD FROM HERE

Logging

Debugging Strategies

Chris Bryan ◆ cbryan@usfca.edu
Department of Computer Science

Motivation

- A debugger is not an option in some settings
  - Multithreading, distributed computing
- Using println() statements are not ideal
  - Easy to create, cumbersome to remove
  - Hard to keep context in large programs.
- Using a custom MyDebug class is not ideal
  - Easy to disable, but inefficient and inflexible

https://www.ibm.com/developerworks/library/os-ecbug/

Apache Log4j 2

- A third-party **library** provided by Apache
- Supports multithreaded or distributed environments
- Efficient and flexible logging implementation
- Configurable without modifying source code

http://logging.apache.org/log4j/2.x/

Simple Log4j2 Example

```
1 Logger log = LogManager.getRootLogger();
2 log.debug("Hello world!");
```

http://logging.apache.org/log4j/2.x/manual/api.html

Logging Levels

- TRACE
  - Used for fine-grained debug messages
  - Usually not shown or used unless really necessary
- DEBUG
  - Used for normal debug messages
  - Most commonly used level for logging

http://logging.apache.org/log4j/2.x/log4j-api/apidocs/org/apache/logging/log4j/Level.html

Logging Levels

- INFO
  - Used for informational messages
  - Often used for major events that were successful
- WARN
  - Used when something concerning happened and there *might* be a problem

http://logging.apache.org/log4j/2.x/log4j-api/apidocs/org/apache/logging/log4j/Level.html

Logging Levels

- ERROR
  - Used when an possibly recoverable error occurred
  - Nearly always shown to the user
- FATAL
  - Used when an irrecoverable error occurred
  - Often used when about to exit prematurely

http://logging.apache.org/log4j/2.x/log4j-api/apidocs/org/apache/logging/log4j/Level.html

Logging Levels

- ALL
  - Turns on all levels of logging, including TRACE
  - Used to turn on all logging for debugging purposes
- OFF
  - Turns off all levels of logging, including FATAL
  - Used to disable logging, speeding up code

http://logging.apache.org/log4j/2.x/log4j-api/apidocs/org/apache/logging/log4j/Level.html

Logging Levels

- TRACE « lowest level, rarely used
- DEBUG « most common
- INFO « informational
- WARN « warnings
- ERROR « errors/exceptions
- FATAL « highest level, rarely used

http://logging.apache.org/log4j/2.0/manual/architecture.html

Configurable Output

- Configuration done via an xml or json file
- Controls what information is output
- Controls which classes output messages
- Controls which levels are output
- Controls where messages are output

http://logging.apache.org/log4j/2.x/manual/configuration.html

Log4j Configuration

- Appenders
  - Controls where log messages are output
  - Commonly the console and/or a log file
- Layouts
  - Control what information is output for an appender
  - Common info: the timestamp, level, and message

http://logging.apache.org/log4j/2.x/manual/architecture.html

Log4j Configuration

- Loggers
  - Identified by a name (often class name)
  - Specify level of message to send to an appender
- Root Logger
  - Default logger, always accessible
  - Uses console appender and outputs ERROR messages by default

http://logging.apache.org/log4j/2.x/manual/configuration.html

Sample Configuration

```
1  <Configuration status="WARN">
2    <Appenders>
3      <Console name="Console" target="SYSTEM_OUT">
4        <PatternLayout pattern="%level: %message %n"/>
5      </Console>
6    </Appenders>
7    <Loggers>
8      <Root level="error">
9        <AppenderRef ref="Console"/>
10     </Root>
11   </Loggers>
12 </Configuration>
```

http://logging.apache.org/log4j/2.x/manual/configuration.html

Pattern Layout

- %level : level of the logging event
  - %level{length=1} to use a single letter
  - %level{lowerCase=true} to convert to lowercase
  - %level{WARN=Warning, …} to change label
- %message : log message
- %n : platform dependent line separator (\n or \r\n)

http://logging.apache.org/log4j/2.x/manual/layouts.html#PatternLayout

Pattern Layout

- %date{pattern} : timestamp for logging event
  - %d{HH:mm:ss:SSS} to output just time
- %thread : thread name (useful later)
- %location : location where logging event created
  - Expensive operation, use with caution
  - See also %logger{}, %class{}, %method, and %line

http://logging.apache.org/log4j/2.x/manual/layouts.html#PatternLayout

Pattern Layout

- Example Pattern
  - [%date{HH:mm:ss:SSS} %-5level{lowerCase=true}] %file@%line %t: %m%n
- Example Output
  - [12:05:53:145 debug] CharacterCounter.java@181 main: Counting characters in file "src".

http://logging.apache.org/log4j/2.x/manual/layouts.html#PatternLayout

Installing Log4j2

- Download latest log4j2* jar files
  - http://logging.apache.org/log4j/2.x/download.html
- Extract and save jar files somewhere convenient
  - Create a directory for all third-party libraries

http://logging.apache.org/log4j/2.x/download.html

Installing Log4j2

- Create a new Java user library in Eclipse
  - Open Eclipse Preferences » Java » Build Path » User Libraries and click "New…"
- Name the library log4j2 in all lowercase with no spaces to avoid Eclipse issues

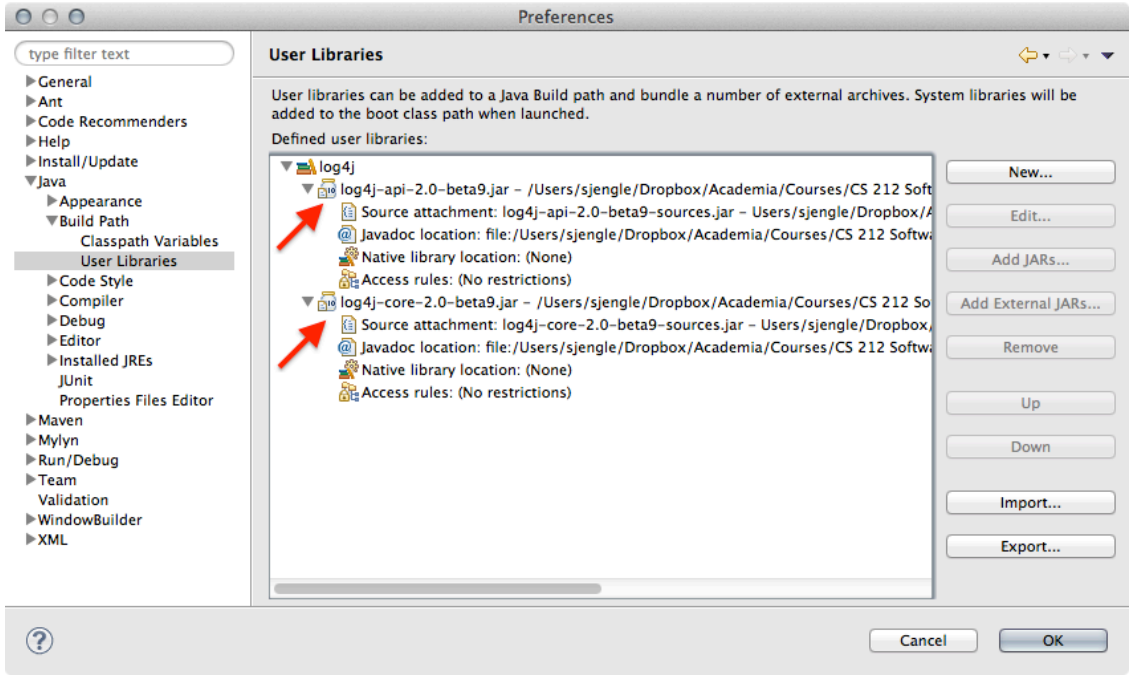
http://tutoringcenter.cs.usfca.edu/resources/adding-user-libraries-in-eclipse.html

Installing Log4j2

- Click "Add External JARs…"
  - Add log4j-api-2.x.jar (version may differ)
  - Add log4j-core-2.x.jar (version may differ)
- Attach source JAR files (optional)
- Attach Javadoc JAR files (recommended)
- Add new user library to build path of project

http://tutoringcenter.cs.usfca.edu/resources/adding-user-libraries-in-eclipse.html

Installing Log4j 2



http://tutoringcenter.cs.usfca.edu/resources/adding-user-libraries-in-eclipse.html

Chris Bryan ◆ cbryan2@usfca.edu
Department of Computer Science